Science of Computer Programming 168 (2018) 18–37

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Non-functional requirements in model-driven development of service-oriented architectures

David Ameller, Xavier Burgués *, Dolors Costal, Carles Farré, Xavier Franch

Departament ESSI, Universitat Politècnica de Catalunya, Jordi Girona Salgado 1-3, 08034 Barcelona, Spain

ARTICLE INFO

Article history:
Received 9 September 2016
Received in revised form 20 July 2018
Accepted 3 August 2018
Available online 8 August 2018

Keywords:
Model-driven development
Non-functional requirements
Quality requirement
Service-oriented architecture
Systematic literature review

ABSTRACT

Any software development process needs to consider non-functional requirements (NFR) in order to deliver a system that complies with its stakeholders' expectations. In a previous mapping study about model-driven development (MDD) for service-oriented architectures (SOA) we found a limited number of approaches managing NFR. The present work aims at analyzing in detail the state of the art in the management of NFR in MDD processes which produce SOA. We have conducted a systematic literature review following a rigorous protocol. We have taken as initial point the mapping study mentioned above and have used the subset of the 31 papers from this study (clustered into 15 approaches) that referred to NFR. We have analyzed them qualitatively in order to answer six research questions. We have built a Software Engineering theory to formalize this analysis. As result we highlight that most of approaches focus exclusively on security and reliability and we observe that NFR are expressed mainly as annotations of functional models represented in UML. From our perspective, existing research on the topic of this study is still scarce and without any evidence of transferability to industry. This situation suggests the need for further investigation efforts in order to produce validated MDD methods capable of generating SOA satisfying NFR stated by stakeholders.

1. Introduction

In the last fifteen years, the study of Service-Oriented Computing (SOC) [1] has attracted a lot of attention from researchers and practitioners. There are several reasons for this success, both purely technological (related to reusability, modularity, etc.) and business-wise (outsourcing of development efforts, pay-per-use, etc.).

As with any other emerging computing paradigm, SOC is not an isolated discipline. Instead, it has to be integrated into the software lifecycle. As a consequence, the relationship of SOC with other orthogonal software engineering streams deserves careful consideration in order to discover opportunities and uncover challenges. In this paper, we are interested in one such stream, namely Model-Driven Development (MDD).

According to Mellor et al., MDD pushes the vision that we can construct a model of a system and then transform it into the real thing [2]. If we apply this description in the context of SOC, the "*model of a system*" is actually a Service-Oriented Architecture (SOA) model, a model that reflects the structure of the SOC application [1]. In fact, OMG released in 2012 a standard named SOA Modeling Language (SoaML [3]) which links SOA and MDD.

* Corresponding author.
E-mail addresses: dameller@essi.upc.edu (D. Ameller), diafebus@essi.upc.edu (X. Burgués), dolors@essi.upc.edu (D. Costal), farre@essi.upc.edu (C. Farré), franch@essi.upc.edu (X. Franch).

https://doi.org/10.1016/j.scico.2018.08.001
0167-6423/

Fig. 1. Models and transformations of MDD.

Some of the authors of this paper participated in a mapping study that described the state of the art of MDD approaches that produce SOAs [4]. One of the objectives of this mapping study was to identify which of these approaches were able to handle Non-Functional Requirements (NFR). From the 129 papers selected in the mapping study, only 31 mentioned some kind of support of NFR management in their MDD approach. The mapping study, however, pointed out the significant influence of NFR in software systems. It also hinted that the approaches were providing in general very limited support to NFR. However, given the descriptive nature of mapping studies, the evidence was rather sketchy and required further investigation in order to be more conclusive. Petersen et al. recommend systematic literature reviews as the natural research instrument for conducting such subsequent research [5].

Following this advice, we present in this paper a Systematic Literature Review (SLR) to delve into the MDD approaches for SOA that support NFR and provide a detailed view of the current state of the art on this particular topic. This SLR is based on the results obtained in the previously mentioned mapping study (i.e., we consider as the starting point the 31 papers out of 129 that mentioned some kind of NFR support in the mapping study). A more in-depth analysis of the approaches is possible in this SLR due to the more manageable number of papers. A greater number of issues can be studied in a more qualitative way and it is feasible to cluster the papers reporting the same proposal in a single group for analysis purposes. Moreover, we make a further step and propose a Software Engineering theory [6] in which we summarize and formalize the results that we have obtained and the analysis that we have made of them. Further research may take advantage of this theory by validating, refining and/or extending its propositions.

The paper is organized as follows: Section 2, the necessary background for the paper; Section 3, the methodology used, including the research questions; Section 4, the results obtained for each research question; Section 5, the discussion of the results; Section 6, the Software Engineering theory that we derive from such a discussion; and Section 7, the conclusions and future work.

2. Background

The three major research topics related to this SLR are described in the next subsections.

2.1. Model-Driven Development

Model-Driven Development (MDD) [2] is a generalization of the OMG's Model-Driven Architecture (MDA) initiative [7], which is proposing since 2000 a comprehensive approach to develop, analyze and reason about software, specifications and related artefacts. Integration and interoperability are the main goals of MDD and the use of models is the key driver to achieve these goals.

Fig. 1 shows the typical MDD process transiting through different models at different levels of abstraction [8]. To structure the presentation, we use the MDA models classification because it is a well-known classification and it provides a common ground to analyze the different MDD approaches found in the literature. The Computational Independent Model (CIM) describes the context and requirements of the system but does not address its structure or processing. This model is transformed into another one, the Platform Independent Model (PIM), through a Model-to-Model (M2M) transformation. Historically in MDA the term "CIM" has been used for a pure business model, one that may need additional system design decisions applied to produce a PIM for a system [7]. The PIM considers the operational capabilities of the application in a platform-neutral way, showing only those parts that can be abstracted out of any platform. Through a subsequent M2M transformation, the Platform Specific Model (PSM) adds to a PIM the details related to the use of a specific platform or set of platforms. Finally, the code (or other development related artefact) is produced by transforming the PSM into a textual representation using a Model-to-Text transformation (M2T). It should be remarked that Fig. 1 shows a conceptual view of MDD, since there may be lots of variations; some typical ones are: absence of CIM; coexistence of different PIMs and/or PSMs; transformations over the same type of models; etc.

Given such an intensive use of models, there is general agreement that metamodeling is an essential foundation for MDD [9]. Defining the corresponding transformation rules, it is possible to relate and to derive a target model from a source one. The extent to which this process is supported by some tool is a fundamental characteristic of every MDD approach.

Besides the classical top-down development, in which the process starts with a CIM and ends with code, there can also be contexts in which the interest is to derive a PIM from another PIM or even from a PSM. In fact, any kind of transformation may be useful, depending on the targeted activity (integration of legacy systems, coordination of services, etc.).

2.2. Non-Functional Requirements

Non-Functional Requirements (NFR) are one of the main research targets of the requirements engineering research community [10]. There are dozens of definitions of the concept of NFR (see [9,11]). For example, they have been defined as:

"(...) global requirements on [the system] development or operational cost, performance, reliability, maintainability, portability, robustness, and the like." [12]. However there is no common agreement on the concrete meaning of the term NFR [10].

Some other terms have been formulated with similar meaning than NFR. Among them, we mention quality requirement (QR), defined as "is a requirement that pertains to a quality concern that is not covered by functional requirements" [13]. For the purposes of this work, we consider the concept of QR as a synonymous of NFR. Even if this decision could be a matter of discussion, it allows increasing the scope of our study and providing a wider view of the state of the art on this topic. Given our interest in SOA, another relevant term is Quality of Service (QoS), defined as "the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" [14].

One main issue in dealing with NFR is their classification into some taxonomy. Software quality attributes (QAs) can be used with this purpose. There are many proposals, being the ISO/IEC 25010 quality standard [15] the most widespread one. We will target NFR as specific requirements of a particular software product, and NFR types (also known as non-functional properties) as general quality aspects of a software product.

2.3. Service-Oriented Architecture

Service-Oriented Architecture (SOA) is a software architectural style that uses services as the main building component [16]. A service, as a software component, is a mechanism to enable access to one or more capabilities [17]. Some of the most distinguishing characteristics of SOA are [16]:

- *Reusability:* Since services are defined as autonomous software components that expose some generic, well-defined functionality, it should be easy to reuse them in a different context or project.
- *Scalability:* Services can be deployed in more than one place. This characteristic brings many opportunities, for instance, the possibility of increasing the number of available services when there is more user demand.
- *Flexibility:* SOA provides means to publish and discover new services. These facilities allow adapting the service compositions of the architecture in order to fulfill some goal at runtime, normally related to some NFR type, e.g., performance. The flexibility of SOA is highly aligned with business principles (e.g., service provider and pay-per-use). This alignment with business is one of the keys to its success.

SOA is a technology-independent architectural style, although the most common implementation of SOA relies on Web Services (the ones adhered to W3C/OASIS protocols [18,19]) and, more recently, on REST Services (also known as RESTful Web Services) [20]. One of the reasons that favors the adoption of MDD approaches to produce SOAs is that developers can reduce the learning curve of the related technologies and standards, for example, in the case of Web Services, developers would have to cope with a huge amount of notations such as WSDL, UDDI, SOAP, XML, WS-Security, WS-BPEL, etc.

2.4. Relevance of the topic

Dealing with NFR is a major challenge in software development. The lack of integration of NFR with functional requirements can result in long time-to-market and more expensive projects [11,21]. Practical impact of NFR has been acknowledged by the research community [16] and has been documented in many studies conducted in industry [22]. Neglecting NFR during software development is a top-ten risk [23], and errors in considering them are the most expensive and difficult to correct [21]. Their influence on the software architecture field is paramount, including architectural design [24], component selection [25] and architecture evaluation [26].

MDD is not an exception to this particular situation. Some mapping studies related to MDD identify NFR-related topics as relevant topics in their studied areas. For instance, Wakil and Jawawi identify "testing & quality" which includes "security" and "QoS" topics as one of the research focuses in model driven web engineering [27]. Meanwhile, Mehmood and Jawawi identify "Code generation from specification of non-functional requirements" as one of the research topics in aspect-oriented model-driven web generation [28]. Being supported or not by MDD approaches, practitioners have to deal with NFR in one way or another but as pointed out in [4,29], only a few MDD approaches include some kind of support for NFR (see Ameller et al. [29] for details on concrete approximations). An ongoing interview-based study with practitioners (whose protocol is reported in [30]) will help to provide more light to the current state of practices at this respect. Last, it is important to remark that we have selected a restricted scope (SOA systems) in purpose. NFR are a general concept to software development, but they can be interpreted and operationalized in different ways depending on a particular software domain or type of system like SOA systems are. We think that opening the scope to other domains would have been an additional and unnecessary threat to the validity of our results.

3. Research methodology

The study conducted in this paper is a Systematic Literature Review (SLR). It has emerged as a result of a previous mapping study performed by some of the authors in the broader topic of MDD for SOA [4]. As reported by Petersen et al. [5], one of the objectives of a mapping study is to call for SLRs that investigate in depth some particular aspect hinted

Table 1
Research questions of the SLR.

RQ1	What types of NFR do MDD approaches for SOA explicitly deal with?
RQ2	Which degree of genericity do the approaches present?
RQ3	What notations do the approaches use for specifying NFR for SOA?
RQ4	Which level of abstraction do the approaches present?
RQ5	What is the role of NFR in the MDD process for SOA?
RQ6	How much evidence is available to adopt the proposed approaches?

in the mapping study. In our case, one of the research questions in the mapping study was "*Do these approaches* [*MDD approaches for SOA*] *deal with NFR?*". The conciseness in the response to this research question in the mapping study was related to the type of analysis used (quantitative) and the fact that only 31 papers mentioned support for NFR. To explore this topic in more depth, in this SLR, we use qualitative analysis (see Section 3.3) and explore additional research questions (see Section 3.1).

In the rest of the section we describe the methodology used for the SLR. We have adopted Kitchenham et al.'s guidelines [31,32] with the particularity that the initial steps of the SLR correspond to the ones in the mapping study.

3.1. Research questions

This SLR addresses six Research Questions (RQ), five on the topic of study and one to assess the maturity of the primary studies. Table 1 lists the six RQs. It is worth to remark that RQ1 and RQ3 were also present in the mapping study, but in this SLR we provide more detailed answers; furthermore, the type of analysis differs from the one used in the mapping study (see Section 3.3).

With RQ1 we intend to have a clear picture of the types of NFR that are being supported by the surveyed MDD approaches. We consider that a given MDD approach supports NFR when 1) they are explicitly specified in some input models and/or 2) are taken into account when generating the output models or code. We will also analyze the terminology used by the researchers to refer to the NFR types and normalize it according to the ISO/IEC 25000 quality standard [15]. This RQ has two dual goals: on the one hand, we will identify in which topics researchers are investing the most effort to support NFR in MDD for SOA; on the other hand, we will identify the types of NFR that are being omitted.

A second aspect of the study is the degree of genericity (RQ2), meaning the flexibility of the MDD approach to deal with NFR. We classify approaches as *generic* when they deal with any type of NFR, and *specific* when they treat a predetermined set of concrete NFR types. The degree of genericity is relevant to understand the current state of the art because it shows how deep or wide the studied approaches are.

RQ3 covers the notation for NFR used in the MDD approaches. With this RQ we want first to record the different notations used in the approaches studied. Also, we will discuss to what extent these notations are expressive (i.e., able to represent different types of NFR), understandable, and reusable (i.e., can be extended/adapted in other contexts or approaches).

Another aspect to consider is the level of abstraction of the NFR (RQ4) when represented with the approach's notation. By level of abstraction we refer to the conceptual distance of the representable NFR from the solution space, in other words, how much operationalized are the representable NFR.

With RQ5 we want to understand how the NFR are used in the MDD processes for SOA. To answer this RQ we will provide a classification of the approaches by the given uses of NFR. In particular, we want to identify at which stage of the MDD process NFR are introduced and how they are incorporated into the solution together with the functional part. We also want to identify how NFR are transformed from model to model.

Finally, in RQ6 we determine the maturity of the approaches by assessing their readiness level based on the evidence provided in the primary studies. The insights to answer this RQ are based on another SLR conducted by Alves et al. [33] together with the analysis of tool availability to support the surveyed approaches.

3.2. From the previous mapping study to the current SLR

Table 2 presents the methodology followed in this SLR that consisted of 8 steps. As mentioned above, the SLR started from an already published mapping study focused on MDD approaches for SOA [4]. Steps 1 to 5 were conducted in the mapping study which included: a selection of venues and digital libraries, a definition of a search string, a set of inclusion/exclusion criteria, and a list of threats to validity. All of these steps apply to the present SLR. Next we provide details on the remaining three steps.

Step 6 selected the set of primary studies for this SLR. We used as starting set the primary studies of the mapping study and then we used the answer to RQ3 of the mapping study ("Do these approaches deal with NFRs?") to filter out those papers that did not mention any NFR support in the MDD process. We found four dominating NFR types: security, reliability, performance and dependability. Some papers deal with NFRs in a generic way and few papers deal with specific NFRs different of the four mentioned above. As a result, from the 129 papers in the mapping study, we kept 31 papers as preliminary set of primary studies for the SLR.

Table 2
From the mapping study to the SLR.

	Step	Number of items
Mapping study	1. Papers obtained from search	1962 papers
	2. Filter by topic	1385 papers
	3. Filter by title and abstract	336 papers
	4. Filter by fast reading	151 papers
	5. Filter by full reading	129 papers
SLR	6. Filter by NFR-aware papers	31 papers
	7. Filter by new exclusion criteria	24 papers
	8. Group by approach	15 approaches

Table 3
List of approaches of the SLR.

Approach	References
A01	[42,51]
A02	[43]
A03	[50]
A04	[59]
A05	[60–62]
A06	[44,47]
A07	[54]
A08	[46]
A09	[53]
A10	[63–65]
A11	[48]
A12	[52]
A13	[55–58]
A14	[49]
A15	[45]

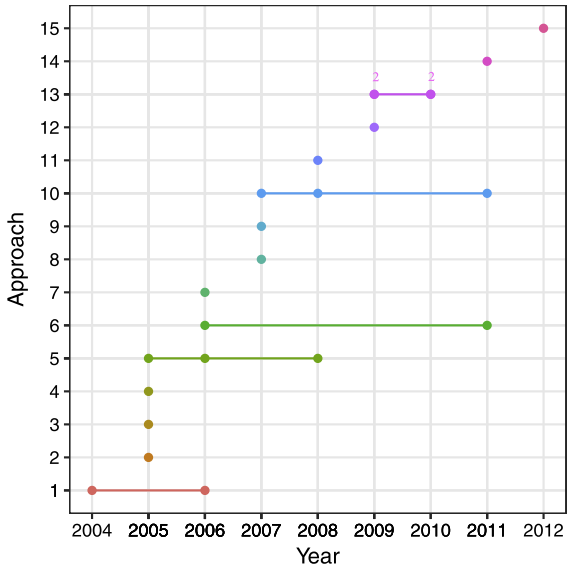

Fig. 2. Approaches and year of publication.

In Step 7 we added a new exclusion criterion, namely: "the contents of the paper should be sufficient to answer the SLR research questions". We included this new exclusion criterion because we found several papers that stated the support of NFR in their approach but no details were provided, therefore we could not perform the deeper analysis required for this SLR. As a result, 7 papers were excluded.

From the set of papers obtained in Step 7, we noticed that many of them belonged to the same researchers and they were reporting the same approach from different points of view or at different stages. Given the qualitative nature of this SLR, we decided to put the emphasis on approaches instead of individual papers, therefore we grouped papers from the same authors. As a result, we clustered the set of primary studies into 5 groups of works and 10 individual works for a total of 15 approaches reported in the SLR (see Table 3 and Fig. 2; the references are listed in the Appendix).

3.3. Analysis strategy

The analysis of this SLR is mainly qualitative. We use qualitative-oriented analysis because this type of analysis works well with small samples [34], as it happens to be the case of this SLR with 15 approaches. Content analysis is one of the most used mechanisms to perform a qualitative analysis. As mentioned in Mayring's book [34], categories are the central point of content analysis and they constitute the central instrument of analysis. The techniques used for content analysis of this study are: summarizing, explication, and structuring.

In particular, the analysis process performed in this study followed two steps. First, we prepared the analysis of the approaches. In this step, we wrote summaries and made presentations (i.e., verbal explication technique) of the approaches in several research meetings. Some preliminary categories were determined in this step. Second, we provided the results for each RQ. These results were obtained applying content analysis techniques such as summarizing and structuring. In this part of the analysis, we determined the final categories for each RQ, and compared their classifications with the ones proposed in the first analysis step. In some cases, we discussed among all researchers the classification of some approach during the meetings.

3.4. Threats to validity

The threats to validity identified in the previous mapping study also apply to this systematic review. In this subsection we include two additional threats and their mitigation strategies:

- *Incorrect grouping of approaches*. The creation of groups from the primary studies may be inaccurate. To mitigate this threat, we applied two steps. First, we grouped papers that had a non-empty intersection of authors, which we considered a necessary condition to be part of the same group. Second, we looked into the details in each group to verify that all of the papers worked over the same approach. As a result, some papers were not included in the same group even if some authors overlapped.
- *Researcher bias*. In addition to the mitigation actions made in the mapping study with regard to the researcher bias, in the SLR the qualitative results could be biased by the background, attitudes and practices of individual researchers. To reduce researchers' bias, the assignment of tasks was made in a way to ensure that all the studied works were analyzed by at least three different researchers (including the steps made for the selection of the primary studies). In addition, we held periodic research meetings for discussion and we followed a strict procedure for editing documents in which all subjective information was annotated with comments and kept traceable until consensus was reached.
- *Incomplete set of primary studies*. As in any literature review, an internal threat to validity is having missed some relevant primary study on the topic of interest. In this literature study, we have not searched for any new primary study after those found in our previous systematic mapping. As main mitigation action, we have made all the details of our protocol fully available to the community, which makes it possible to update the set of primary studies and analyze the validity of our findings after such update.

4. Results

This section summarizes the results found in this study.

4.1. RQ1: types of NFR

The most evident result obtained for RQ1 is the high amount of approaches that address security: all except 3 approaches provide some kind of support for security. Among the security aspects that are supported by these approaches, the most usual ones are those related to authentication and authorization (e.g., "*we discuss the possibility of denoting the security properties of access control, data integrity, and confidentiality*"; A12 [52]), and also those related to communications channels and their encryption (e.g., "*the language is tailored to model cryptographic protocols and supports* (...) *several cryptographic operations such as encryption, decryption, digital signatures*"; A15, [45]). The second most recurrent type of NFR supported is reliability (in some cases is referred as fault tolerance, e.g., "*Fault Tolerance, the number of avoided fault patterns compared with the number of fault patterns to be considered*"; A08, [46]) followed by availability which was mentioned in three approaches (see Table 4 and Fig. 3). In the case of A09, the paper refers to dependability, but the authors clarified that "*the term dependability means both system availability and reliability*" [53]. NFR types related to performance were mentioned also in three cases (mentioned as efficiency in A03; mentioned as response time, resource utilization, and throughput in A08).

Some of the approaches use the term QoS which may be more aligned with the SOA context. In these cases, they normally refer to the NFR types that can be measured on execution time (e.g., availability, response time, etc.), for example, "*the catalogue of characteristics and dimensions that describe the QoS of web services*" (A06, [47]), or "*the means to specify system properties related to the Quality of Services*" (A07, [54]). In Table 4, when we state "OMG QoS" we mean that apart from using the term QoS to refer to the NFR types, they also advocate for the OMG standard in QoS [35].

One of the approaches (A14) claims generic NFR support (as reported in Section 4.2), although only security, performance, and reliability are fully addressed (see RQ2).

Table 4
Results RQ1.

Approach	Security	Reliability	Availability	Performance	QoS
A01	X				
A02	X	X			
A03	X			X	OMG QoS
A04	X				
A05	X				
A06	X	X	X		OMG QoS
A07		X			QoS
A08		X	X	X	OMG QoS
A09		X	X		QoS
A10	X	X			
A11	X				
A12	X				
A13	X				
A14	X	X		X	OMG QoS
A15	X				

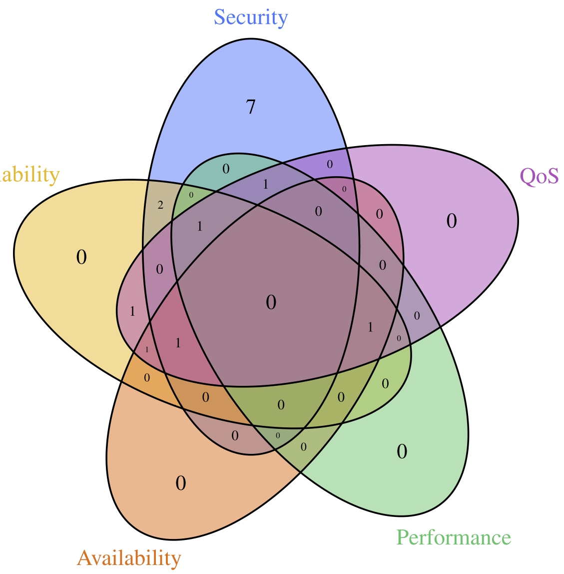

Fig. 3. Venn diagram (RQ1).

4.2. RQ2: degree of genericity

Almost all approaches deal with specific NFR types. The exception is A14 which presents NFR support for security, performance, and reliability, but also defines a specific notation to extend the approach to other NFR types ("*The approach is based on a profile for modeling services in UML, called UML4SOA* [...] *which we extend to include very generic non-functional specifications which are bound 'per contract' to the services in the structural model*", [49]). Such genericity is provided by a metamodel which has one or more NFR types (named *NFCharacteristics* in A14) defined each one by one or more dimensions (*NFDimensions*). In this way the software engineer can specify the required NFR types, e.g., *Security*, tagged as *NFCharacteristic* as well as their relevant *NFDimensions* (e.g., *Encryption*, *DigitalSignature*). It must be noted, though, that transformation rules have to be designed specifically for the different NFR types ("*we created PIM2PSM and model-to-code transformations to facilitate service development for reliable and secure middleware. These transformations currently handle reliable message communication and security in service-oriented systems*" and "*Its modular implementation allows for future extension in other non-functional domains* (*e.g.*, *logging*) *and other service platforms* (*e.g.*, *SCA*) *as well*").

The rest of the approaches are specific to a range of predetermined NFR types (see Table 5 and Fig. 4). This means that the approaches exploit some knowledge (in the form of metamodel, technique, semantics, etc.) specific of 1 to 3 types of NFR (see Table 4). It is worth to remark that some of them can be extended to support other types of NFR (even that they do not necessarily provide the facilities included in A14). This is explicitly mentioned by some approaches (e.g., "*Additional characteristics about, e.g.*, *service utilization, integrity, scalability or accuracy, can be defined or existing ones be modified or removed in a similar way*", A06, [47]). The extension mechanism to apply is usually based on the extension of the NFR specification languages and the definition of new transformation rules, as explicitly mentioned by one of the approaches ("*Because of the huge diversity of non-functional properties, applications may require new non-functional properties and constraints that are not*

Table 5
Results RQ2.

Approach	Generic/specific	Extensible
A01	Specific	Yes
A02	Specific	
A03	Specific	
A04	Specific	
A05	Specific	
A06	Specific	Yes
A07	Specific	
A08	Specific	Yes
A09	Specific	
A10	Specific	Yes
A11	Specific	
A12	Specific	
A13	Specific	Yes
A14	Generic	Yes
A15	Specific	

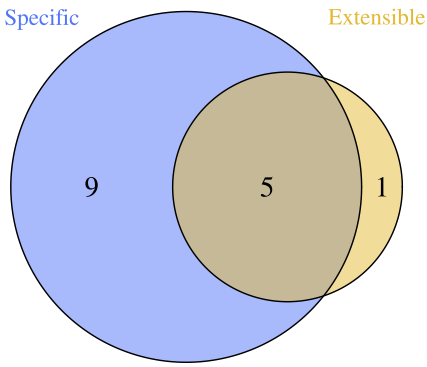

Fig. 4. Venn diagram (RQ2).

supported in the current proposed MDD framework. In order to introduce new non-functional properties and constraints, application developers are required to extend FM-SNFPs, UP-SNFPs and transformation rules", A10, [64]).

The last column of Table 5 reports whether the method provided by the approach is extensible to deal with other NFR types. Approach A14, although generic, is classified as extensible, too. The reason is that its notation is generic while the transformations applied by the proposed method must be designed specifically for different types of NFR.

4.3. RQ3: NFR notations

The most recurrent observation of the studied approaches is that most of them use a graphical notation (i.e., use of diagrams that represent concepts, entities, relationships, attributes, constraints, interactions, etc.). Only one approach (A07) uses a textual notation, a "*Domain Specific Language (DSL) based on the SOA paradigm, which is a specialization of the* [*ArchWare*] *ARL language*" [54]. Among the graphical notations, UML is the most used (see Table 6). Only one approach (A12) uses a graphical notation that is not based on UML. In this latter case, flowchart-like diagrams describing business processes are enriched with special "symbols" to represent confidentiality and data-integrity constraints.

UML-based approaches can be classified into three main groups according to the UML extension mechanism that they use (see Table 6 and Fig. 5):

- *Plain models*. No UML extensions are used. The metamodels are represented as UML class diagrams and the models themselves are instance diagrams.
- *Lightweight extensions*. New stereotypes are defined using UML Profiles.
- *Heavyweight extensions*. The notation includes a well-defined metamodel.

Here we must single out three approaches, A02, A04 and A13 whose classification requires some remarks. The notation in A02 uses basic annotations on UML activity diagrams to define "*security zones and boundaries*" [43]. Therefore, it may be argued that these annotations are similar in purpose and syntactic appearance to stereotypes. In the case of A04, the approach does not define its own notation but sketches some examples or just suggests taking advantage of notations from the previous literature to, for instance, "*model security policies in UML using profiles, metamodels, constraints, or other*

Table 6
Results RQ3.

Approach	Base notation language	UML extension
A01	UML	Plain
A02	UML	Lightweight
A03	UML	Lightweight
A04	UML	Lightweight/Heavyweight
A05	UML	Lightweight
A06	UML	Plain
A07	ARL (textual DSL)	N/A
A08	UML	Heavyweight
A09	N/A	N/A
A10	UML	Heavyweight
A11	N/A	N/A
A12	Flow chart	N/A
A13	UML	Plain/Lightweight
A14	UML	Heavyweight
A15	UML	Lightweight

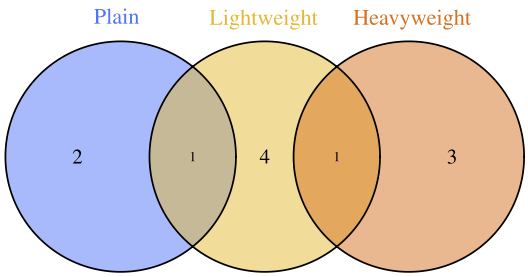

Fig. 5. UML extensions.

mechanisms" [59]. Approach A13 consists of several papers proposing two different notations, one is based on plain UML models and the other one is a lightweight extension to UML.

Finally, two approaches (A09 and A11) do not provide any kind of notation to directly represent the NFR but provide guidelines or patterns to realize them. In the case of A09, NFR are not represented, instead "*specific failures of components and web services and their handling strategies*" [53] are required. A11 addresses "*the non-functional requirements separately from the functional requirements by embodying the security knowledge in the form of security patterns*" [48]. Therefore A09 and A11 have been excluded from RQ3.

4.4. RQ4: level of abstraction

To facilitate the analysis of this research question, we have defined three possible levels of abstraction for the supported NFR:

- *High abstraction*. The NFR represented are totally unaware of the solution. For example, a NFR that states that "*the system shall have a highly secured login mechanism*".
- *Intermediate abstraction*. The NFR represented are aware of the some aspects related to the particular architecture or domain, but they do not go into the technical details. For example, a NFR that states "*the login service shall use encrypted communications channels*".
- *Low abstraction*. The NFR represented are very tied to the concrete solution and its implementation, including technical details. In fact, they can be considered as a design decision. For example, a NFR that states "*the LDAP service shall only allow SSL connections*".

Most of the approaches surveyed (see Fig. 6) represent the NFR at a high level of abstraction (e.g., "*A software architect applies security requirements in abstracted form to a service model*", A05, [62]; "*In the QoS Profile, a QoS characteristic is a quantified aspect of the QoS* [...] *which is defined independently of the means by which it is represented, managed or controlled*", A06 [47]). One common aspect among all the approaches that are considered of high abstraction, is that the NFR notation refers to a characterization of the desired QoS of the involved services (see Table 7).

Only two approaches are placed in the intermediate level of abstraction (A01 and A02). Both require knowledge of the particular problem to specify the NFR. In A01 the classes are complemented with OCL constraints that reference other classes of the domain, therefore the approach needs knowledge of the domain model to represent the NFR. In A02 the dis-

Table 7
Results RQ4.

Approach	Abstraction	Required knowledge
A01	Intermediate	Domain, Class attributes, ID, etc.
A02	Intermediate	Architecture, Boundaries, etc.
A03	High	Desired characteristics of the services
A04	High	Desired characteristics of the services
A05	High	Desired security: integrity, authentication, etc.
A06	High	Desired characteristics of the services
A07	High	Desired characteristics of the services
A08	High	Desired characteristics of the services
A09	N/A	N/A
A10	Low	Technical knowledge about the desired solution
A11	N/A	N/A
A12	Low	Technical knowledge about the desired solution
A13	Low	Technical knowledge about the desired solution
A14	High	Desired characteristics of the services
A15	Low	Technical knowledge about the desired solution

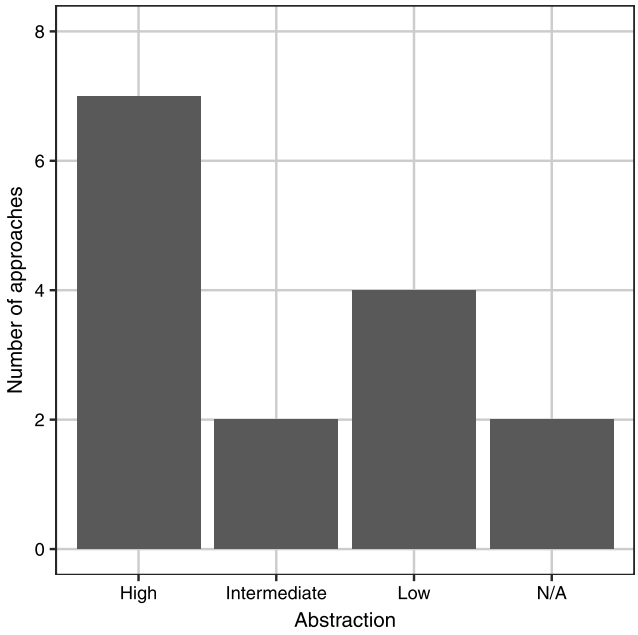

Fig. 6. Number of approaches per abstraction level.

tribution of the services is noted with security requirements, therefore the approach needs knowledge about the deployment architecture to note the NFR.

In the approaches classified as of low abstraction, the NFR notation was very close to the solution that operationalizes the NFR. For example, "*an Indexing Service requires an X.509 certificate*" (A10) [65], "*Bank applies the stereotype* $<< TLS + SSA >>$. *This means TLS with server side authentication*" (A15, [45]). In A12 and A13 this low level of abstraction was hidden: by asking for the concrete technologies to implement the NFR during the MDD process (A12) and by the definition of "profiles" that included the concrete technologies used for the solution (e.g., "*X509-Token*", A13 [58]).

The approaches that did not provide any NFR notation (A09 and A11) were not classified according to the level of abstraction.

4.5. RQ5: NFR in the MDD process

The approaches can be grouped into three types (see Fig. 7) with respect to the way that NFR participate in the process. In all the cases, there is a top-down transformation of models, starting at higher levels of abstraction that, in most cases (but not all), matches the CIM-PIM-PSM transformations. NFR appear at the early stages of the process in the three types, but we see differences on the way in which they are held and the moment in which they are plugged into the system. The first type of approaches (leftmost diagram) attach the NFR information to the models of the MDD process while the second one (middle diagram) creates a series of models which represent NFR that run in parallel to the functional model until they

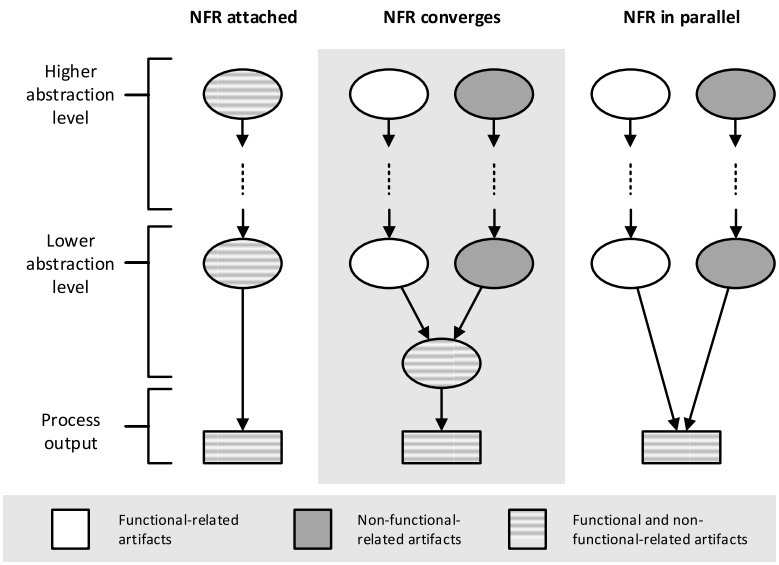

Fig. 7. Models and transformations of MDD.

converge at some point. In the third typology (rightmost diagram) the sequences of models do not converge but each one ends contributing to the final product.

This typology of transformation is used to classify the approaches (see Table 8 and Fig. 8, leftmost area). In most cases, NFR and functionalities run together in the models of the MDD process, in a few cases they start separated before converging and only in one case they run in parallel and at the end the final SOA is complemented or enhanced with some NFR-specific output ("*we introduced a model-driven approach to generate security policies*", A13 [55]).

We also show in Table 8 and Fig. 8 (middle area) how NFR participate in the process, either through transformations or acting as quality constraints. The model transformations are in most cases specific to the supported types of NFR ("*It is significant to observe that security policies are specified and refined throughout the life cycle, undergoing transformations from one phase to the next*", A04, [59]); the remaining approaches use the NFR as parameters of the transformations or use the QoS properties to select the candidate services. Two approaches, A07 and A13, propose some additional models to the classical CIM, PIM, and PSM levels. These are the models that take into account architecture issues in the process ("*the approach described here attempts to combine both formal architecture-centric and model-driven paradigms*", A07) which is not surprising to us as we stated in [29]. Actually, these approaches deal with grid architectures that are considered as a special case of service architecture ("*we consider the Grid as a SOA and provide the means to specify system properties related to the QoS*" [54]).

We show in Table 8 and Fig. 8 (rightmost area) the life-cycle activity supported by the approaches. Their usual target is to contribute to the development of the final system but in a few of them the target is to monitor the QoS ("*In this paper we present an implementation of a Model Driven Architecture (MDA) based framework for the runtime monitoring of QoS properties*", A08, [46]). In either case, NFR appear at the very first stage of the process, guiding the development of the system or the monitoring tool.

Concerning the development of the system, the majority of approaches focus on the creation of new services complying with the NFR. However, two of them (A03, A06) are instead oriented to the selection of the component services to be included in a service orchestration by filtering the candidates according to the desired QoS which is monitored. In both cases, QoS-annotated workflow models are constructed and algorithmically evaluated to find the most desirable alternative. Approach A03 can address different QoS properties (execution time, price, user satisfaction, etc.) whereas A06 only focuses on service reliability. Finally, in approach A14 service orchestrations are also the object of "*performance estimates and reliability analysis using the stochastically timed process algebra PEPA as the underlying analytical engine*" [49].

4.6. RQ6: approaches maturity

To analyze the maturity of the approaches we have assessed the kind of evidence provided in the primary studies according to the evidence categories used in another SLR conducted by Alves et al. [33]:

- No evidence.
- Evidence obtained from demonstration or working out toy examples.
- Evidence obtained from expert opinions or observations.
- Evidence obtained from academic studies, e.g., controlled lab experiments.
- Evidence obtained from industrial studies, e.g., causal case studies.
- Evidence obtained from industrial practice.

Table 8
Results RQ5.

Approach	NFR attached	NFR converges	NFR in parallel	Specific transformation	Parameter transformation	QoS selection	Development	Monitoring
A01	X			X			X	
A02	X			X			X	
A03	X					X	X	
A04	X			X				X
A05	X			X			X	
A06	X					X	X	
A07		X			X		X	
A08	X			X				X
A09	X			X			X	
A10	X				X		X	
A11	X			X			X	
A12		X			X		X	
A13			X	X			X	
A14		X		X			X	
A15	X			X			X	

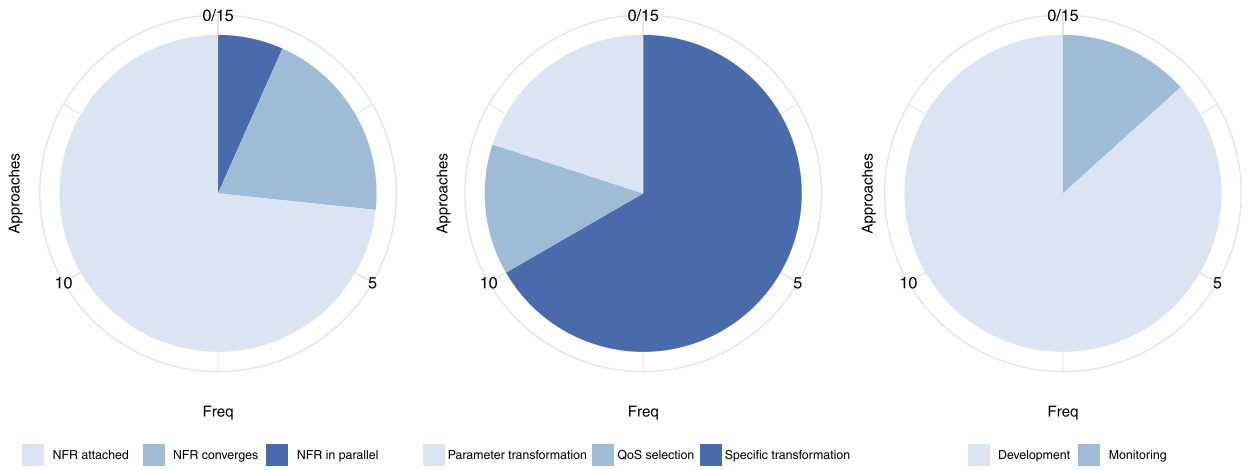

Fig. 8. Classification of NFRs in the MDD process.

Furthermore, in the case of MDD, another aspect that clearly indicates the readiness level is the availability of a tool implementing the proposed approach. This information will be also used to answer RQ6. The following two subsections describe the results found related to tool support and evidence provided, respectively.

4.6.1. Tool support

Eight out of the fifteen approaches report some kind of tool support while three mention that it is still work in progress (e.g., "*As the implementation of the security model view is yet still work in progress, we are not able to provide a sufficient evaluation on the model's capabilities*", A12, [52]), see Table 9. One of the approaches reporting tool support states that it is limited to only a subset of the functionality ("*WS-Security and WS-Policy standards are not available in Biztalk*", A02, [43]). In addition, three other approaches remark that the reported tool is a prototype (e.g., "*We have a tool prototype for MDS and Pattern based configuration*", A05, [62]) and none of the approaches reports to have developed a commercial tool. In general, we observe that there is a lack of details about the tool support in the studied approaches.

4.6.2. Evidence

The evidence provided by all the approaches but one consists in presenting toy examples and, remarkably, one of these toy examples is extracted from a case study of an EU project ("*This example has been taken from one of the case studies of the Sensoria project*", A14, [49]), see Table 9. The exception is A10 that includes, in addition to a toy example, an empirical evaluation of the proposal ("*Empirical evaluation results show that BALLAD significantly reduces the burdens/costs to implement and maintain non-functional properties in service-oriented applications. BALLAD and FM-SNFPs are designed and implemented efficient and scalable*" [64]). However, none of the approaches provides evidence obtained either from industrial studies or industrial practice, although this is sometimes mentioned as part of the future work (e.g. "*Future work will necessarily encompass further validation of the approach presented against larger projects. We plan to apply it to more complex case studies in collaboration with industry*", A14, [49]). By contrast, regarding to the industry affiliation of the authors involved in the approaches, we have found that almost half of the approaches involve at least one author from industry (see Table 9 and Fig. 9, last column).

Table 9
Results RQ6.

Approach	Tool support	Evidence provided	Authors from industry
A01	No (work in progress)	From toy examples	Yes
A02	Yes (for a subset of the functionality)	From toy examples	Yes
A03	No	From toy examples	Yes
A04	No	From toy examples	Yes
A05	Yes (prototype)	From toy examples	Yes
A06	Yes (prototype)	From toy examples	No
A07	Yes	From toy examples	No
A08	No (work in progress)	From toy examples	No
A09	Yes (prototype)	From toy examples	Yes
A10	Yes	From controlled lab experiments	Yes
A11	No	From toy examples	No
A12	No (work in progress)	From toy examples	No
A13	No	From toy examples	No
A14	Yes	From toy examples	No
A15	Yes	From toy examples	No

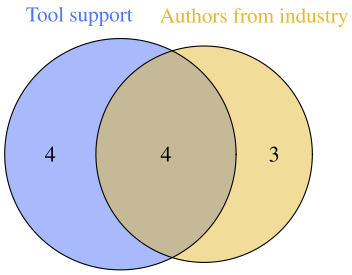

Fig. 9. Tool support and authors from industry.

5. Discussion

In this section we search for possible explanations to the results observed.

5.1. RQ1: types of NFR

Results of RQ1 show that most of the approaches that handle NFR deal with security issues. In fact, the number of approaches could be considered even greater if we adhere to the view of some authors that include availability as a principle that supports security [36].

This result is in contrast with a recent survey on a similar topic [37], in which software architects from industry mentioned dependability and performance as more important than security when architecting service-based systems. One possible way to understand this apparent contradiction is that there has been a great effort in the standardization of security related issues for SOA systems, which has resulted in a bunch of proposals like WS-Security, WS-Trust, etc. For an MDD approach, having a well-defined starting point is crucial in order to define metamodels, transformations rules, etc. making the approach easier to implement. This interpretation is aligned with the observation made by de la Vara et al. who mention type of projects as one of the influencing factors on the relative importance of NFR types [38].

Another possibility for the discrepancy with [37] could be that academic interests and practitioners needs are not aligned, but other facts contradict this possibility: a) even considering the different frequency, the three top NFR types in the afore mentioned survey [37] are the same top NFR types in this SLR (understanding dependability as reliability and availability); b) as already commented, 7 out of the 15 approaches involved practitioners in the research team and 6 out of these 7 approaches dealt with security (it remains open to know if these practitioners also played the role of software architects).

Another interesting way to read the results is the dual one, i.e., NFR types such as usability, portability, and maintainability are omitted. Our interpretation is that this is a consequence of the type of software system addressed. Usability is disregarded because SOA systems, implementing service-based systems, normally do not include a user interface. On the other hand, portability is an inherent characteristic of services, and maintainability is an inherent characteristic of MDD.

5.2. RQ2: degree of genericity

The results obtained for RQ2 are clearly biased towards a low degree of genericity, i.e. they tend to be specific of one particular type of NFR. This result already emerged in the review included in [29]. The reason may be that NFR types are very heterogeneous, making it difficult to provide a uniform specification and treatment for them. This observation aligns

with the observations by Whittle et al. [39] from practitioners' approaches to MDD, which point out that successful adoption of MDD is based on focused solutions in terms of system domain, scope and notation used.

Nevertheless, a careful analysis of the surveyed proposals leaves the door open to more promising results in terms of generalization because some of them claim to be extensible to the support of additional NFR types (usually by extending the NFR specification languages and adding new transformations). The adoption of the MDD approach contributes to well-structured frameworks for the proposals mainly due to its foundation in metamodeling and model transformations, e.g. it provides a clear separation of concerns between the specification of the NFR and their impact on model transformations. The existence of standards related to MDD such as UML which is extensible itself facilitates the addition of new NFR types; for example, A10 explicitly mentions: "*UP-SNFPs is built on the UML standard metamodel with the standard extension mechanism, and application developers can add stereotypes and tagged-values representing new non-functional properties*" [23].

One more thing to consider is that when taking these academic approaches into practice, practitioners will need support for more than one type of NFR in their project, therefore a combination of MDD approaches will be needed. It is not clear (for the studied approaches) if this combination is possible, but surely it would not be smooth.

5.3. RQ3: NFR notations

We remark the fact that most notations to represent NFR are based on UML. At first this seems to be in sharp contrast with the findings by Whittle et al. [39] which report that in their study MDD practitioners created or used languages specific for their domain (DSLs), rather than using general-purpose languages such as UML. However, it should be also pointed out that only a small fraction of notations that we have reviewed in this study use "plain" UML models and that some of the DSLs reported in [39] are expressed as UML profiles. Moreover, UML is also the most used notation to model functionality as we stated in the mapping study in which this SLR relies. Other quite widespread notations are used in a few works only. BPMN, for instance, is the notation found in only 5 (out of 129) papers. One of them [64] is NFR-aware and did pass the filter in step 6.

One possible reason for this dominance of UML to represent NFR is that in many of the approaches reviewed in our study, NFR are not intended to be used to generate NFR-aware functional code but to create different types of complementary and necessary artefacts: deployment descriptors, policy specifications, etc. Those artefacts are typically XML documents and, in this respect, UML to XML standards (e.g. XMI, XSLT) and tools are quite mature and widespread.

Moreover, UML is a broadly accepted notation for software modeling, beyond MDD, as it is reported by Forward et al. [40]. In that study, "communicate to others" and "readability" are found to be the two most appreciated attributes of a modeling tool. That evidence supports our view that the use of UML as core representation mechanism fosters the understandability of the proposed notations.

Regarding the expressivity and reusability of the proposed notations, we have found that those approaches that use a well-defined metamodel are the ones that are more able to represent different types of NFR and/or can be easily adapted/extended to represent them.

A final thought on the use of UML as a base notation for representing NFR: a stronger common ground on which different approaches could define their own UML extensions is missing. This common ground should natively provide the common abstractions and constructs that we find in SOA-based applications: services, capabilities, contracts, etc. SoaML [3], whose first definite version was published later than any of the approaches considered here, could be a good candidate for this "common ground" for future approaches. An interesting follow-up of this SLR would be to report on the adoption of SoaML by the community.

5.4. RQ4: level of abstraction

We found out that the level of abstraction for the representable NFR is polarized, meaning that the approaches tend to be either in a very abstract or in a very concrete notation. In our opinion, this happens because of the inherent nature of NFR. On the one hand, NFR can be embraced in an abstract way (e.g., the goals that are sought for the SOA system) but at the cost of losing the detail and the control of the implementation. One particular risk of this way of handling NFR is that the MDD approach becomes the "golden hammer" (i.e., everything is shaped by the same pattern). In this situation the approach may reduce the amount of work of the developers, but at some point developers will need to manually adapt the "general solution" to the specific characteristics of the software product. On the other hand, NFR can be operationalized, meaning that the MDD approach will be narrowed to concrete known and suitable solutions, but unable to evolve or adapt to new ones. Handling NFR as operationalized solutions helps to produce fully operative systems, but developers become unaware of the reasoning behind the decisions made by the MDD approach. One possible way to reconcile both options is to embed a goal-refinement approach into the MDD process. This view has not been found in any of the approaches surveyed in this paper.

We think that handling NFR in an abstract way is the best solution for SOA-based systems because services can be seen as atomic elements isolated from the technology (e.g., when accessing to an external service only the public functionalities are visible). In this situation, what is necessary from the NFR point of view, is the possibility to characterize these services (one particular notation that seems to recognize this is the OMG QoS [35] used by A03, A06, A08, and A14).

In a different case, we found approaches in which NFR are embedded in the approach (A09, and A11). In these approaches NFR are not noted in the models, instead there is a set of implicit NFR used in a holistic way in all the parts of the system. In these approaches there is no control at all on how the NFR are applied, and they cannot be considered abstract or operationalized because they are not present in the models. Still these approaches can be considered useful in highly specialized contexts. Again we may draw some analogy to Whittle et al.'s study [39] in which they conclude that practitioners tend to use model-driven approaches in focused domains; in this context, some NFR may be considered recurrent and thus built-in in the MDD process' core.

5.5. RQ5: NFR in the MDD process

We showed in Table 8 how most approaches considered NFR attached to functional models instead of having independent models for them. We think that there are two main reasons that may justify this trend: first, while there are a plethora of models for representing functionality (especially when working with UML), this is not the case for NFR, therefore most authors may prefer to consider NFR as annotations in the functional model; and second, maintaining two models in parallel for merging them at some step of the MDD process is challenging. An additional reason is that detailed NFR often refer to particular functions of the system, therefore it may be natural to express this link at the modeling level. A consequence of this dominance of this annotation-based approaches is the difficulty in obtaining reusable solutions on NFR transformations due to the intertwining of their treatment with the functional models.

A similar situation appears with respect to model transformation, where one approach (specific transformation) dominates the others. We again refer to Whittle et al. observation on the specificity of approaches [39]: creating a particular transformation delivers more down-to-earth solutions at the price of sacrificing genericity. It is worth to mention how the development-from-scratch view seems to be still more popular than reusing services (which intuitively should be a dominant case in the context of service-orientation), according to the scarcity of proposals that consider service selection in the MDD process.

In the extreme, results of the SLR show how development is still the dominant perspective for the great majority of proposals. This is not surprising and in fact aligns to the general observation made in our previous mapping study on MDD in SOA [4].

5.6. RQ6: approaches maturity

The evidence provided by the approaches is placed on the weak end of the hierarchy from [33] since it is mainly based on the presentation of toy examples and none of the approaches provides stronger kinds of evidence such as industrial studies or, even better, industrial practice. This is consistent with the fact that the tool support for the approaches is also weak (mostly prototypes, work in progress or no tool support reported) even if effective tool support is a precondition for transferring the approaches to the industry, e.g., the empirical study reported in [41] finds that the maturity of the tools is an important determinant for the adoption of MDE. This evidence weakness indicates that the proposals are still not well-established and need to be improved to be applicable in real settings.

At the same time, a positive fact is that almost half of the approaches involve practitioners as authors. This is quite a high share of approaches. For instance, it is significantly higher if we compare to the mapping study on MDD in SOA [4]. This may point out to the interest of practitioners when NFR come to play, as we already reported in [37] where a significant majority of respondents stated that NFR are equally important or even more important than functional requirements for architects developing SOA systems. We think that this interest by industry demands for additional research on the topic.

We have compared the results of the present SLR with those of its preceding mapping study [4], and in particular the issues that refer to the kind of evidence provided, tool support and involvement of authors from industry. The most remarkable observations are:

- *Kind of evidence provided*. On the one hand, 16.3% of the papers from the mapping study provided evidence from industrial studies while none of the NFR approaches from the SLR provides this kind of evidence. On the other hand, the only kind of evidence provided by all except one of the NFR approaches (93.3%) were toy examples while this was the case of only 64.3% of the mapping study papers.
- *Tool support*. We do not notice much difference on this issue since 53% of the NFR approaches report some kind of tool support and it was reported in 59.7% of the papers from the mapping study.
- *Involvement of industry authors*. Nearly half of the NFR approaches involve authors from industry (7 out of 15) while only 31.8% of the papers from the mapping studies do.

These results may indicate that, although NFR are considered interesting from the practitioner side, the approaches that deal with them have not reached the same level of maturity, at least according to the kind of evidence provided, as the general MDD approaches for SOAs have.

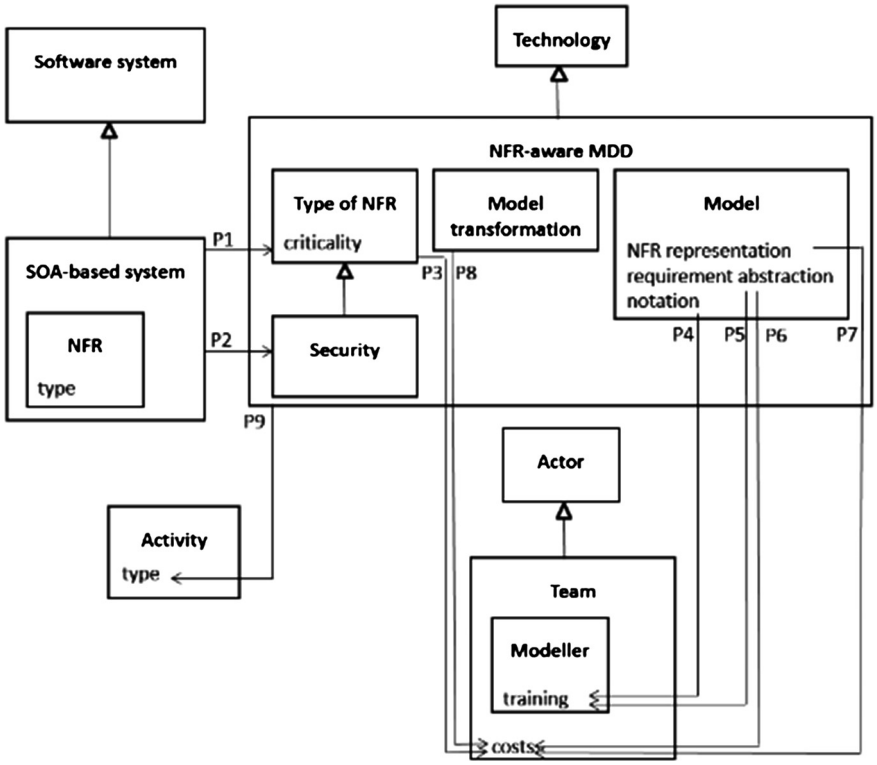

Fig. 10. Theory constructs in a class diagram.

6. Theory model

In this section, we consolidate the results and analysis from the survey in the form of a theory for software engineering following the methodology provided by Sjøberg et al. [6]. The theory may be viewed as the starting point for further research that may consolidate the findings of this study, propose new ones or refute some of our propositions. Please note that Sjøberg et al.'s methodology consists of five steps, but for the sake of brevity we present the results in a more abridged way. We draw our conclusions including suggestions for further research in Section 7.

Sjøberg et al. build a theory by first introducing its main constructs either as specializations of one of four archetype classes (Actor, Technology, Activity and Software Systems) or as attributes inside these new specialization classes; and then asserting propositions as relationships among constructs. Explanations provide the justifications to the propositions. The theory is summarized in Fig. 10 and Table 10 and explained below.

In the diagram depicted in Fig. 3, we follow the UML-inspired notation proposed by Sjøberg et al. [6]. Constructs are represented as classes or attributes of a class. Classes are drawn as boxes. A class may be a subclass (denoted by the UML generalization arrow, e.g. *Security* is a subclass of *Type of NFR*) or a component class (drawn as a box within another box, e.g. *Type of NFR* is a component of *NFR-aware-MDD*). The arrows in the diagram connect the sources and the targets of the propositions.

6.1. Constructs

- *Software System*. It presents the scope of the study. We are focusing on one particular type of system, namely *SOA-based systems*. As any other system, a particular SOA-based system satisfies some *NFR* that belong to some *Type of NFR*.
- *Technology*. The main focus of the paper. In our case, we are evaluating the *NFR-aware MDD* technology. Being NFR-aware, it is important to know the *Types of NFR* addressed by the method; we introduce the *criticality* attribute to recognize the fact that some types may be more important than others for a particular proposal. As in any MDD-based approach, *Models* lie in the very heart of the method. According to our study, the main attributes to consider in a method are: the *notation* used to represent the models; the way in which NFR are represented (*NFR representation*); and the *requirement abstraction* level in this representation. Being MDD, it is also important to know which types of *Model transformations* are applied.
- *Actor*. The main actor in the theory is the *Team* that will build the SOA-based system. A relevant attribute is their *cost* (we will be interested in knowing the impact of the NFR-aware methods in the overall cost of the team). In the team

Table 10
Theory constructs, propositions and explanations.

Constructs	
C1	*NFR-aware MDD*
C2	*SOA-based system*
C3	*Type of NFR* (identifier of a category of similar NFR according to the ISO/IEC 25010 classification schema)
C4	*Security* (a particular type of NFR, defined as in the ISO/IEC 25010 standard)
C5	*Criticality* (perceived importance of an NFR type in a given MDD method)
C6	*Costs* (perceived cost for a team applying a given MDD method)
C7	*Model* (a representation of reality)
C8	*Modeller* (an IT professional who writes models)
C9	*Model notation* (type of modeling language, including *Visual* and *UML* as values)
C10	*Training* (time required for modeller to master a new method)
C11	*Requirement abstraction* (level of abstraction in which a requirement is expressed, including as extreme values *goal* and *operationalization*)
C12	*NFR representation* (strategy for representing NFR with respect to the functional part of the model, including *annotations* as value)
C13	*Model transformation* (strategy for applying the MDD process over the model that represents the NFR, including *specific* as value)
C14	*Activity* (including *development* and *monitoring* as values)
Propositions	
P1	Not all types of NFR have the same criticality when developing SOA-based systems with NFR-aware MDD
P2	Security is the most critical type of NFR when developing SOA-based systems with NFR-aware MDD
P3	Dealing with more than one NFR type in NFR-aware MDD for SOA-based systems increases costs
P4	The use of visual notations in representing models involved in NFR-aware MDD processes for SOA-based systems decreases modellers' training
P5	Expressing NFR as goals decreases training in NFR-aware MDD for SOA-based systems
P6	Expressing NFR as operationalizations decreases costs in NFR-aware MDD for SOA-based systems
P7	Representing NFR as annotations decreases costs in NFR-aware MDD for SOA-based systems
P8	Defining specific transformations for NFR decreases costs in NFR-aware MDD for SOA-based systems
P9	NFR-aware MDD for SOA-based systems focus mainly on development activities. In some cases it supports monitoring activities or NFR-evaluation of different design alternatives.
Explanations (explanation E*k* explain proposition P*k*)	
E1	SOA systems are a family of software systems which some particularities. Qualities as security, performance and availability are extremely important, while others as usability are less important or even do not apply
E2	In MDD, the existence of standards makes easier to developed models shared by a community, and security has several well-defined standards in the context of SOA systems
E3	NFR are very different from each other, and dealing with them with the same models and techniques is hard and requires more human effort
E4	Since natural language is not an easy option for MDD approaches, the next natural option for modellers is a visual notation, over more formal representations like logic which could facilitate more sophisticated techniques but are not easy to use by these modelers. Nowadays, UML is by far the most used visual notation in software engineering.
E5	Goals are close to the modelers' universe of discourse, therefore modellers may find easier to adapt the proposed method
E6	Operationalized requirements are close to the final artefact to be delivered in the MDD process, therefore the method is easier to implement
E7	Maintaining two simultaneous models for functional and non-functional requirements for merging them later is conceptually challenging
E8	In concordance to P3 (E3), the conceptual diversity of NFR types from each other makes difficult to provide generic transformation rules, and customized rules are preferred instead
E9	In general, MDD approaches are used mainly for software development, and SOA-based systems are not an exception. In SOA-based systems, monitoring is a fundamental activity for ensuring service level agreements (which are mainly stated over NFR types). NFR can also be analyzed on different design alternatives using analytic/formal models to produce feedback to designers for improving the quality of the software development models.

we identify a crucial actor in MDD approaches, the *Modeller*, for whom we need to know the *training* required by the method at hand.
- *Activity*. We just identify the *type* of activity.

6.2. Propositions

We provide the facts surveyed in this study that support the propositions, organized by research questions. The text links to the propositions P*k* which are detailed in Table 10.

- *RQ1*. The great majority of approaches deal with security requirements, especially related to authentication, authorization and encryption (P2). Reliability is the second most supported type, whilst availability and performance are mentioned by a few approaches. Several types are not even mentioned (P1).
- *RQ2*. Almost all approaches deal with a specific NFR type, except for one which claims to be NFR-type generic through the use of a metamodel-based solution (P3).
- *RQ3*. Almost all approaches use a visual notation to represent the model (P4). UML dominates with all the visual modeling approaches (except one) using it. UML-based approaches range evenly from plain models to lightweight and heavyweight extensions.
- *RQ4*. Most of the approaches either represent NFR at a high level of abstraction (more as goals than as requirements) or at a low level of abstraction (being almost an operationalization of the requirement), but not in the middle (P5, P6).
- *RQ5*. In reference to the role of NFR in the MDD process:

- *Relation to the functional model*. A majority of approaches attach NFR as annotations to the functional part, with a few approaches that start with both models separated and merge them at some later moment (P7).
- *Model transformation*. A majority of approaches propose specific transformations for the supported types of NFR, with a few of them using NFR either as parameters of such transformations or as input for a selection of services to be integrated in an SOA (P8).
- *Life-cycle activity*. NFR are addressed in relation to different types of activities (P9):
  - Almost all approaches are conceived to support system development.
  - A couple of approaches consider NFR for monitoring activities.
  - Few approaches evaluate the NFR for different design alternatives by building and analyzing formal models with the goal of selecting the best alternatives and/or improving the quality of the target system.

7. Conclusions

In this systematic literature review we surveyed the state of the art in the management of NFR in MDD processes used in the context of SOA-based applications. We designed and followed a rigorous protocol (based on a former mapping study [4]) which uncovered up to 15 proposals to answer the different research questions that we identified. We analyzed the findings and formulated a theory to consolidate them and organize the knowledge for future research on the topic.

As result of the work presented in this paper, we propose five important aspects to consider in future NFR-aware MDD approaches.

- *Completeness of the NFR support*. We have observed that security is the most attractive type of NFR when addressing NFR in MDD for SOA, but we also found approaches dealing with reliability, availability and performance. Furthermore we can argue that types of NFR such as compatibility, maintainability and portability are inherently supported by MDD, SOA, or both. In addition to this, usability can be considered irrelevant in the case of SOA because the user interface is normally not part of the designed software system. Therefore we say that a full spectrum of types of NFR are supported in one way or another when using MDD for SOA. However, we cannot say to what extent these types of NFR are supported by these approaches. Our position is that further studies are needed to measure this in more detail, but we believe that the approaches that use quality standards (e.g., OMG QoS) to represent the NFR are better prepared to deal with a full extent of diversity when dealing with NFR.
- *Extensibility of the MDD approach*. In this study, we found only one approach that claims to have generic support for different types of NFR and several that claim to be extensible, but the majority of approaches provide support for specific types of NFR and no means of extension. Being computer science constantly evolving, approaches that do not support any kind of extension have great chances of becoming obsolete in a short time. Real-life projects usually require to deal with a wide range of different types of NFR and, from our point of view, an MDD approach that only addresses one or two NFR types while neglecting most of them will hardly appeal to practitioners.
- *Integrability of the MDD approaches*. The MDD approaches studied in this study are those focused on handling NFR, however NFR are only one dimension of the problem. Other aspects such as the concrete functionalities or the support of specific technologies may be provided by a different MDD tool or framework, therefore the capability of integration of the MDD approach that supports the NFR is a must. To facilitate this integration, a possible strategy is to use UML. On the one hand, most MDD approaches are based on UML, and on the other hand, integrating models from different languages would be cumbersome (if even possible). Furthermore, from the integrability point of view, it seems advisable to extend by lightweight UML extensions. Lightweight extensions are normally limited to the description of several stereotypes for the UML standard elements, in this situation it seems feasible to combine two or more MDD approaches (each one using their own stereotypes). On the contrary, the case of approaches with heavyweight extensions (additions or modifications to the UML metamodel) are difficult to integrate because some of the integrated approaches will be unable to understand certain parts of the models.
- *High (or low) NFR abstraction*. In RQ4 we analyzed the types of NFR supported from the perspective of the level of abstraction. We observed that the majority opted for a higher abstraction but there are also several approaches (especially the most recent ones) that opted for a lower abstraction. In this case, we see benefits and drawbacks on both sides, therefore the best solution needs to be pondered for each situation. On the one hand, higher abstractions favors the reuse of the approach among different software projects but difficult the approach maintainability because of its complexity. On the other hand, lower abstractions are less complex, and therefore easy to maintain, but since the addressed NFR are very concrete in general it should be difficult to reuse the approach for different projects.
- *Lack of evidence*. The lack of evidence is an indicator of low maturity. In this study we checked the existence of tool support that implements the proposed approach and we also checked if the approach was used on a case study. The results are not encouraging: most approaches do not provide a tool (which should be considered a must in the context of MDD) and especially concerning, none of the approaches was validated in industrial settings. Our opinion is that the research community should make an effort to attract the industry participation to validate the proposed approaches, and this, of course, cannot be done without an implemented tool.

Funding: This work has been partially supported by the Spanish MICINN [project TIN2016-79269-R].

References

[1] M.P. Papazoglou, D. Georgakopoulos, Introduction: service-oriented computing, Commun. ACM 46 (10) (2003) 24–28.
[2] S.J. Mellor, A.N. Clark, T. Futagami, Guest editors' introduction: Model-driven development, IEEE Softw. 20 (5) (September 2003) 14–18.
[3] OMG, Service oriented architecture modeling language (SOAML), version 1.0.1, http://www.omg.org/spec/SoaML/1.0.1/PDF, 2012.
[4] D. Ameller, X. Burgués, O. Collell, D. Costal, X. Franch, M.P. Papazoglou, Development of service-oriented architectures using model-driven development: a mapping study, Inf. Softw. Technol. 62 (2015) 42–66.
[5] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08, British Computer Society, Swinton, UK, 2008, pp. 68–77.
[6] D.I.K. Sjøberg, T. Dybå, B.C.D. Anda, J.E. Hannay, Building theories in software engineering, in: F. Shull, et al. (Eds.), Guide to Advanced Empirical Software Engineering, Springer, 2008.
[7] MDA Guide revision 2.0, http://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf.
[8] M. Brambilla, J. Cabot, M. Wimmer, Model-Driven Software Engineering in Practice, Morgan & Claypool Publishers, 2012.
[9] C. Atkinson, T. Kühne, Model-driven development: a metamodeling foundation, IEEE Softw. 20 (5) (September 2003) 36–41.
[10] M. Glinz, On non-functional requirements, in: IEEE RE 2007, pp. 21–26.
[11] L. Chung, J.C. Sampaio do Prado Leite, On non-functional requirements in software engineering, in: Conceptual Modeling: Foundations and Applications, Springer, 2009, pp. 363–379.
[12] J. Mylopoulos, L. Chung, B.A. Nixon, Representing and using nonfunctional requirements: a process-oriented approach, IEEE Trans. Softw. Eng. 18 (6) (June 1992) 483–497.
[13] K. Pohl, C. Rupp, Requirements Engineering Fundamentals, Rocky Nook Inc., 2011.
[14] ISO IEC, ISO 8402:1994 – Quality management and quality assurance – Vocabulary, 1994.
[15] ISO/IEC 25010, System and Software Engineering Systems and Software Quality Requirements and Evaluation (Square): System and Software Quality, 2010.
[16] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.
[17] C.M. MacKenzie, K. Laskey, F. McCabe, P.F. Brown, R. Metz, Reference model for service oriented architecture 1.0, OASIS Standard, http://docs.oasis-open.org/soa-rm/v1.0/, 2006.
[18] W3C: Web Services Activity, http://www.w3.org/2002/ws/.
[19] OASIS: Standards, https://www.oasis-open.org/standards.
[20] T. Erl, B. Carlyle, C. Pautasso, R. Balasubramanian, SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST, Prentice Hall Press, 2012.
[21] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos, Non-functional Requirements in Software Engineering, Kluwer Academic, 2000.
[22] D. Ameller, C. Ayala, J. Cabot, X. Franch, Non-functional requirements in architectural decision-making, IEEE Softw. 30 (2) (2013) 61–67.
[23] B. Lawrence, K. Wiegers, C. Ebert, The top ten risks of requirements engineering, IEEE Softw. 18 (6) (2001) 62–63.
[24] S. Kim, D. Kim, L. Lu, S. Park, Quality-driven architecture development using architectural tactics, J. Syst. Softw. 82 (2009) 1211–1231.
[25] X. Franch, J.P. Carvallo, Using quality models in software package selection, IEEE Softw. 20 (2003) 34–41.
[26] M.R. Barbacci, M.H. Kleiin, C.B. Weinstock, Principles for Evaluating the Quality Attributes of a Software Architecture, Technical Report, SEI CMU, 1997.
[27] K. Wakil, D.N.A. Jawawi, Model driven web engineering: a systematic study, e-Informatica Softw. Eng. J. 9 (1) (2015) 87–122.
[28] A. Mehmood, D.N.A. Jawawi, Aspect-oriented model-driven code generation: a systematic mapping study, Inf. Softw. Technol. 55 (2013) 395–411.
[29] D. Ameller, X. Franch, J. Cabot, Dealing with non-functional requirements in model-driven development, in: IEEE RE 2010, pp. 189–198.
[30] D. Ameller, X. Franch, C. Gómez, J. Araujo, R.B. Svensson, S. Biffl, J. Cabot, V. Cortellessa, M. Daneva, D.M. Fernández, A. Moreira, H. Muccini, A. Vallecillo, M. Wimmer, V. Amaral, H. Brunelière, L. Burgueño, M. Goulão, B. Schätz, S. Teufl, Handling non-functional requirements in model-driven development: an ongoing industrial survey, in: 23rd IEEE International Requirements Engineering Conference (RE), 2015, pp. 208–213.
[31] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K.E. Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Trans. Softw. Eng. 28 (8) (2002) 721–734.
[32] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report, 2007, http://www.dur.ac.uk/ebse/resources/Systematic-reviews-5-8.pdf.
[33] V. Alves, N. Niu, C. Alves, G. Valença, Requirements engineering for software product lines: a systematic literature review, Inf. Softw. Technol. 52 (8) (August 2010) 806–820.
[34] P. Mayring, Qualitative content analysis. Theoretical foundation, basic procedures and software solution. Social science open access repository, http://nbn-resolving.de/urn:nbn:de:0168-ssoar-395173, 2014.
[35] OMG, UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, August 2003.
[36] National Institute of Standards Technology, An Introduction to Computer Security: The NIST Handbook, Special Publication 800-12, October 1995.
[37] D. Ameller, M. Galster, P. Avgeriou, X. Franch, A survey on quality attributes in service-based systems, Softw. Qual. J. 24 (2015) 271–299.
[38] J.L. de la Vara, K. Wnuk, R.B. Svensson, J. Sánchez, B. Regnell, An empirical study on the importance of quality requirements in industry, in: SEKE 2011.
[39] J. Whittle, J. Hutchinson, M. Rouncefield, The state of practice in model-driven engineering, IEEE Softw. 31 (3) (2014).
[40] A. Forward, O. Badreddin, T.C. Lethbridge, Perceptions of software modeling: a survey of software practitioners, in: C2M:EEMDD@ECMFA 2010, pp. 12–24.
[41] P. Mohagheghi, W. Gilani, A. Stefanescu, M.A. Fernandez, An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases, Empir. Softw. Eng. 18 (1) (2013) 89–116.

Appendix: SLR References

[42] M. Alam, R. Breu, M. Breu, Model driven security for web services (MDS4WS), in: INMIC 2004: 8th International Multitopic Conference, Proceedings, 2004, pp. 498–505.
[43] R. Anzbock, S. Dustdar, Semi-automatic generation of web services and BPEL processes – a model-driven approach, in: W.M.P. Van der Aalst, B. Benatallah, F. Casati, F. Curbera (Eds.), Business Process Management, Proceedings, 2005, pp. 64–79.
[44] P. Bocciarelli, A. D'Ambrogio, A model-driven method for describing and predicting the reliability of composite services, Softw. Syst. Model. 10 (2) (2011) 265–280.
[45] M. Borek, N. Moebius, K. Stenzel, W. Reif, Model-driven development of secure service applications, in: Proceedings of the 2012 IEEE 35th Software Engineering Workshop (SEW 2012), 2012, pp. 62–71.
[46] K. Chan, I. Poernomo, QoS-aware model driven architecture through the UML and CIM, Inf. Syst. Front. 9 (2–3) (2007) 209–224.
[47] A. D'Ambrogio, A model-driven WSDL extension for describing the QoS of web services, in: ICWS 2006: IEEE International Conference on Web Services, Proceedings, 2006, pp. 789–796.

[48] N.A. Delessy, E.B. Fernandez, A pattern-driven security process for SOA applications, in: S. Jakoubi, S. Tjoa, E.R. Weippl (Eds.), ARES 2008: Proceedings of the Third International Conference on Availability, Security and Reliability, 2008, pp. 416–421.
[49] S. Gilmore, L. Goenczy, N. Koch, P. Mayer, M. Tribastone, D. Varro, Non-functional properties in the model-driven development of service-oriented systems, Softw. Syst. Model. 10 (3) (2011) 287–311.
[50] R. Gronmo, M. Jaeger, Model-driven methodology for building QoS-optimised Web service compositions, in: L. Kutvonen, N. Alonistioti (Eds.), Distributed Applications and Interoperable Systems, 2005, pp. 68–82.
[51] M. Hafner, M. Alam, R. Breu, Towards a MOF/QVT-based domain architecture for model driven security, in: O. Nierstrasz, J. Whittle, D. Harel, G. Reggio (Eds.), Model Driven Engineering Languages and Systems, Proceedings, 2006, pp. 275–290.
[52] M. Jensen, S. Feja, A security modeling approach for web-service-based business processes, in: 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, Proceedings, 2009, pp. 340–347.
[53] M. Jiang, Z. Yang, A model-driven approach for dependable software systems, in: A. Mathur, W.E. Wong, M.F. Lau (Eds.), USIC 2007: Proceedings of the Seventh International Conference on Quality Software, 2007, pp. 100–106.
[54] D. Manset, H. Verjus, R. McClatchey, F. Oquendo, A formal architecture-centric model-driven approach for the automatic generation of Grid applications, in: ICEIS 2006: Proceedings of the Eighth International Conference on Enterprise Information Systems – Information Systems Analysis and Specification, 2006, pp. 322–330.
[55] M. Menzel, C. Meinel, A security meta-model for service-oriented architectures, in: 2009 IEEE International Conference on Services Computing, 2009, pp. 251–259.
[56] M. Menzel, C. Meinel, SecureSOA modelling security requirements for service-oriented architectures, in: 2010 IEEE International Conference on Services Computing (SCC), 2010, pp. 146–153.
[57] M. Menzel, I. Thomas, C. Meinel, Security requirements specification in service-oriented business process management, in: 2009 International Conference on Availability, Reliability, and Security (ARES), vols. 1 and 2, 2009, pp. 41–48.
[58] M. Menzel, R. Warschofsky, C. Meinel, A pattern-driven generation of security policies for service-oriented architectures, in: 2012 IEEE 19th International Conference on Web Services, 2010, pp. 243–250.
[59] N. Nagaratnam, A. Nadalin, M. Hondo, M. McIntosh, P. Austel, Business-driven application security: from modeling to managing secure applications, IBM Syst. J. 44 (4) (2005) 847–867.
[60] Y. Nakamura, M. Tatsubori, T. Imamura, K. Ono, Model-driven security based on a Web services security architecture, in: 2005 IEEE International Conference on Services Computing, Proceedings, vol. 1, 2005, pp. 7–15.
[61] F. Satoh, Y. Nakamura, K. Ono, Adding authentication to model driven security, in: ICWS 2006: IEEE International Conference on Web Services, Proceedings, 2006, pp. 585–592.
[62] F. Satoh, M. Tatsubori, Y. Nakamura, N.K. Mukhi, K. Ono, Methodology and tools for end-to-end SOA security configurations, in: IEEE Congress on Services 2008, Proceedings, Pt. I, 2008, pp. 307–314.
[63] H. Wada, J. Suzuki, K. Oba, A feature modeling support for non-functional constraints in service oriented architecture, in: L.J. Zhang, W. Vander Aalst, P.C.K. Hung (Eds.), IEEE International Conference on Services Computing, Proceedings, 2007, pp. 187–195.
[64] H. Wada, J. Suzuki, K. Oba, Leveraging early aspects in end-to-end model driven development for non-functional properties in service oriented architecture, J. Database Manage. 22 (2) (2011) 93–123.
[65] H. Wada, J. Suzuki, K. Oba, A model-driven development framework for non-functional aspects in service oriented architecture, Int. J. Web Serv. Res. 5 (4) (2008) 1–31.